10-2004

Flexible Verification of MPEG-4 Stream in Peer-to-Peer CDN

Tieyan LI
Institute for Infocomm Research

Yongdong WU
Institute for Infocomm Research

Di MA
Institute for Infocomm Research

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

DOI: https://doi.org/10.1007/978-3-540-30191-2_7

Flexible Verification of MPEG-4 Stream in Peer-to-Peer CDN

Tieyan Li[1], Yongdong Wu[1], Di Ma[1], Huafei Zhu[1], Robert H. Deng[2]

[1] Institute for Infocomm Research (I^2R)
21 Heng Mui Keng Terrace, Singapore 119613
{litieyan, wydong, madi, huafei}@i2r.a-star.edu.sg
[2] School of Information Systems
Singapore Management University
469 Bukit Timah Road, Singapore 259756
{robertdeng}@smu.edu.sg

Abstract. The current packet based stream authentication schemes provide effective and efficient authentication over a group of packets transmitted on erasure channels. However, by fixing the packets in transmission, any packet manipulation will cause authentication failure. In p2p content delivery network where a proxy-in-the-middle is able to store, forward, transcode and transform the stream, previous schemes are simply unapplicable. To address the problem, we propose a flexible verification scheme that relies on special stream formats (i.e. Unequal Loss Protection ULP scheme [7]). We apply the so called Unequal Loss Verification ULV scheme into MPEG-4 framework. The encoding, packing, amortizing and verifying methods are elaborated in this paper. Our analysis shows that the scheme is secure and cost effective. The scheme is indeed content aware and ensures the verification rate intuitively reflecting a meaningful stream. Further on, we describe the general method of publishing and retrieving a stream in p2p CDN.

1 Introduction

Peer-to-peer Content Delivery Networks (p2p CDNs) are emerging as the next generation network architecture [10]. This overlay networks not only enable static files to be published, stored, shared and downloaded easily and reliably (e.g. [25]); but even make real-time streams broadcasted on your PCs (e.g. SplitStream [12] and CoopNet [13]). The end users are experiencing innovative p2p applications and benefiting more and more from the widely adopted CDN architectures. While effective and efficient delivery is the desirable features, security issues like authentication, integrity and confidentiality are more important issues that must be considered seriously. Our study in this paper concentrates on stream authentication.

Stream authentication schemes [14–24] have been intensively studied. Most of them assume an erasure channel such as Internet where packets are lost from time to time. And packet loss increases the difficulty of authenticating streams.

Packet loss has different reasons: router discards packets due to network congestion; receiver discards packets when it has no enough resources; proxy discards unimportant content intentionally so as to meet the network and device requirements. To deal with the problem, a body of works [23, 24] used erasure codes [11] to tolerate arbitrary patterns of packet loss. However, in our p2p CDN setting where a packet can be manipulated, these schemes are simply unapplicable.

This paper introduces the transcoding and transforming operations by a proxy-in-the-middle. The proxy, in the p2p CDN setting, behaves more like a gateway on application layer who can store, reorganize, forward and modify the received packets. It has a more active role in delivery than a router working on network layer and simply forwarding packets. Our work relies on traditional packet based authentication schemes with signature amortization. Based on special stream structure and packaging method, we analyze the stream encoding methods and propose a flexible verification scheme. The scheme allows packet-manipulation by proxies. It can verify in many ways, extend easily and scale well. In p2p CDN, we can also publish the stream as well as its authentication data in a reliable way. Our analysis shows that the scheme is secure and cost-effective. Briefly, we summarize our main contributions as follows:

- We study packet based stream authentication schemes and identify their fixed-packet problem which makes them unable to be used in packet manipulation scenarios such as in p2p CDN.
- We propose an Unequal Loss Verification ULV scheme and apply it into MPEG-4 framework. The scheme is flexible, extensible and scalable.
- We introduce a general method on how to publish, republish and retrieve a stream in p2p CDN. The method could be used practically and transparently.

Paper organization: Section 2 states the problem of traditional packet based authentication scheme. We then define the generic model in section 3. In section 4 we elaborates the core operations in our ULV scheme. Following on, we analyze the security and performance issues in section 5. We propose in section 6 the publishing method in p2p CDN. In section 7 we compare some related approaches. At last, we conclude our paper and point out our future tasks.

2 Problem statement

Of all the authentication schemes [14–24], Gennaro and Rohatgi [15] proposed off-line and online protocols for stream signature using a chain of one time signatures. Their method increases traffic substantially and can not tolerate packet loss. Wong and Lam [16] used a Merkle hash tree over a block of packets and signed on the root of the tree. Each packet can be authenticated individually by containing the signature and the nodes in tree to reconstruct the root. Perrig et al. [18] proposed Time Efficient Stream Loss-tolerant Authentication (TESLA) and Efficient Multi-chained Stream Signature (EMSS) schemes. The schemes are based on symmetric key operations, which uses delayed disclosure of symmetric keys to provide source authentication. The publisher is required online for

disclosing the keys. Miner and Staddon [21] authenticated a stream over lossy channel based on hash graph, but their scheme is not scalable. The traditional schemes can be categorized as hash graph-based [21], tree-based [20, 16] and symmetric key-based [18]. Other approaches deploy erasure codes to resist arbitrary packet loss. Park et al. [23] described an authentication scheme SAIDA by encoding the hash values of packets and the signature on the whole block with information dispersal algorithm. By amortizing the authentication data into the packets, the method reduces the storage overhead and tolerates more packet loss. Recently, Pannetrat et. al. [24] improved SAIDA by constructing a systematic code to reduce the overhead. All of above schemes are Packet based Stream Authentication Schemes (P-SASs).

In typical P-SAS setting, the packets are prepared by the producer and delivered to the receiver via an erasure channel. Along the channel, each packet is processed as an atomic units. The intermediates (e.g. routers) perform only store and forward functions. The packets are either dropped or lost, but not modified due to any reason. P-SASs work well in this sender-receiver (S-R) model. However, in p2p CDN setting, a proxy performs a more active role since it can not only store and forward packets, but also transcode and transform the packets. More security problems are rendered in this Sender-Proxy-Receiver (S-P-R) model.

Transcoding mechanism is provided in Fine Granular Scalability (FGS) [3] to distribute a MPEG-4 stream efficiently and flexibly over heterogeneous wired and wireless networks [4–6]. The transcoding mechanism allows a proxy to discard data layers from the lowest priority layer to higher layers until the resource restrictions are met. The packet size is reduced accordingly. This transcoding strategy differs from packet dropping strategy. Because the transcoded stream can tolerate the same number of packet loss as the original stream, the error-resilience capability is not decreased. Thus, a receiver is able to verify authenticity of the packet origin even if the stream is transcoded.

However, transcoding can not enlarge the packets. In certain network conditions, enlarging packets does help reduce the packet loss rate. It is not difficult to find out the relationship between packet loss rate vs. bit-rate vs. packet size [9]. Under a low bit-rate (e.g. 50kB/s), decreasing the packet size will incur the decreasing of packet loss rate. But in high bit-rate (e.g. 200kB/s), increasing the packet size will cause the same effect that is desirable. To fit in fluctuant network conditions, transforming is necessary and at least as important as transcoding.

3 The general model

Fig. 1 sketches the basic model. It consists of three parts: preparation by source, modification by proxy and verification by receiver. We describe them as follows:

Part 1: Preparation The producer encodes the video objects according to the MPEG-4 standard. The producer prepares the packets for the object group

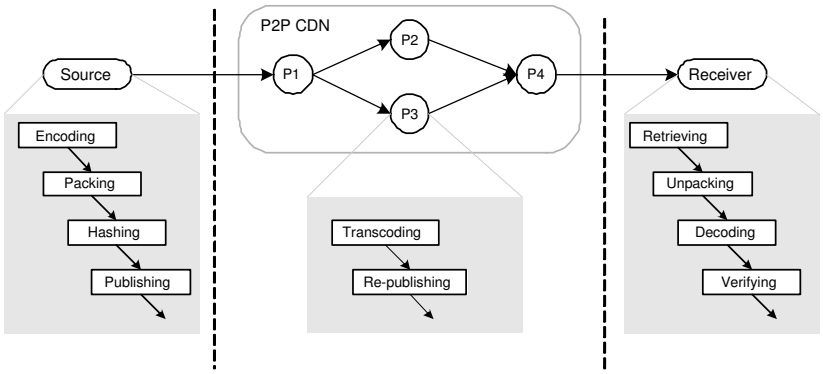

Fig. 1. A general model of publishing and verifying a stream. It depicts three main parts: *Part 1-preparation*, *Part 2-modification* and *Part 3-verification*

based on the priorities of the video objects and layers. The producer generates authentication data including signature and integrity units. The authentication data is amortized over the group of packets. The protected stream is then published on certain p2p CDN and ready to be delivered.

Part 2: Modification To meet the requirement of the network bandwidth or the end device resource, proxies may transcode or transform the stream without affecting verification of the stream. The proxy needs to republish the modified stream in p2p CDN.

Part 3: Verification This part is actually reversing the preparation part. The receiver retrieves a stream as well as its authentication data from the p2p CDN. It then unpacks, decodes the packets. With recovered authentication data, the receiver can verify the signature and integrity. [1]

4 Unequal loss verification ULV scheme

According to [1, 2], a MPEG-4 presentation is divided into sessions including units of aural, visual, or audiovisual content, called media objects. A video sequence (or group, denoted as VSs) includes a series of video objects (VOs). Each VO is encoded into one or more video object layers (VOLs). Each layer includes information corresponding to a given level of temporal and spatial resolution, so that scalable transmission and storage are possible. Each VOL contains a sequence of 2D representations of arbitrary shapes at different time intervals that is referred to as a video object plane (VOP). Video object planes are divided further into macroblocks (MBs) of size 16×16. Each macroblock is encoded into six blocks $B_1, B_2, \cdots, B_6$ of size 8×8 when a 4:2:0 format is applied. In a virtual object sequence VS, VOs, VOLs, VOPs, MBs and Blocks are arranged based

[1] Common assumption is that the verification is conducted non-interactively, we have the same sense, yet still define an interactive scenario in section 6.3.

on a predefined style. Refer to [7] for details on Unequal Loss Protection (ULP) scheme.

In Fig.2, we illustrate a typical hierarchical object tree in one visual object group of a MPEG-4 stream. Based on the tree structure, we are able to generate the hash values bottom-up. At last, a merkle hash tree is formed and its root is to be signed by the originator as the commitment. The tree structure shown in Fig.2 is based on the priority levels of various objects, layers as well as the planes. For instance, The top object VO_1, as the fundamental layer, has the highest priority over the whole object group. The lower the level an object stays in the tree, the lower the priority it has. However, according to different applications, the tree can be constructed adaptively. By signing once on the root of the MHT, the originator actually commits a whole virtual object group to the receivers. Suppose a stream consists of n virtual object groups, n signatures are to be generated to authenticate the stream. Hereafter, we use authentication data to represent both the signatures and the hash values. We discuss how to publish authentication data for each group in a reliable way in next section. Follows, we elaborate the procedures of generating the hash values and signatures. Then, the authentication data is to be amortized into the packets using existing information dispersal algorithm [23]. Transcoding and transforming operations towards the object group are allowed in transmission. We show that we can verify the group in either case at last.

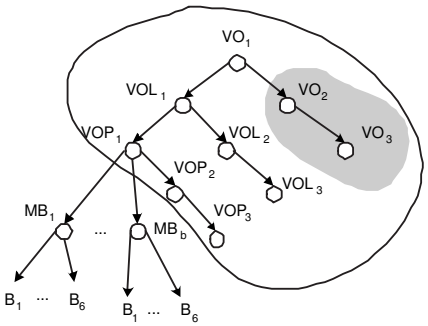

Fig. 2. A typical tree structure of an object group with priority levels from VO_1, VO_2, ... down to $VOLs$, $VOPs$, MBs and $Blocks$. Only the hash values circled out as a partial tree will be taken in authentication data. The shadow part that covers subtrees (VO_2-VO_3) will be removed to demonstrate the transcoding operation in section 4.3

In this paper, we frequently use tools like Merkle hash tree (MHT) [8] and erasure correction coding (ECC), as well as some notations listed in table 1.

4.1 Generating authentication data

Above we defined an object group tree structure, we now generate the MHT recursively from the leaf nodes (e.g. the blocks) to the root. At the bottom

Table 1. Notations

m	A pre-image, a message
$h(.)$	A collision resistant hash function such as SHA-1
K_s	The private key of the producer
K_p	The public key of the producer
Sign	The signature algorithm: $\sigma = \mathsf{Sign}(K_s, m)$, such as RSA
Veri	The verification algorithm: $\mathsf{Veri}(K_p, \sigma, m)$ output $\{true, false\}$

layer, we compute hash values of Blocks with equation 1

$$h_{B_i} = h(Block_i \parallel i), i \in \{1, 2, \cdots, 6\} \tag{1}$$

The hash values of the macroblock MB_j is

$$h_{MB_j} = h(h_{B_1} \parallel h_{B_2} \parallel \cdots \parallel h_{B_6}) \tag{2}$$

where h_{B_i} $(1 \leq i \leq 6)$ is the hash value of block $Block_i$ in macroblock MB_j,

Upward, one upper layer node N with a set of child nodes $C = (C_1, C_2, ..., C_c)$, we compute the hash value of N by the following equation 3:

$$h_N = h(C_1 \parallel C_2 \parallel \cdots \parallel C_c) \tag{3}$$

According to different layers, also refer to Fig. 2, the formulas for calculating the MHT hash values are

$$Level_1 - (Blocks) : L1_j = h(B_j \parallel j), j \in \{1, 2 \cdots, 6\} \tag{4}$$
$$Level_2 - (MBs) : L2_j = h((L1_1 \parallel \cdots \parallel L1_6)), j \in \{1, 2 \cdots, b\} \tag{5}$$
$$Level_3 - (VOPs) : L3_j = h((L2_1 \parallel \cdots \parallel L2_b) \parallel L3_{j+1}), j = 1, 2, \cdots \tag{6}$$
$$Level_4 - (VOLs) : L4_j = h(L3_1 \parallel L4_{j+1}), j = 1, 2, \cdots \tag{7}$$
$$Level_5 - (VOs) : L5_j = h(L4_1 \parallel L5_{j+1}), j = 1, 2, \cdots \tag{8}$$

where (4) computes the hash of each block, (5) calculates the hash of each block and (7)(or (8)) calculates, respectively, the hashes of each object layer and object recursively.

Finally, the object group hash is given as:

$$h_G = h(L5_1 \parallel G \parallel ID) \tag{9}$$

The producer now signs the group hash h_G using its private key K_s and gets the signature as:

$$\sigma = \mathsf{Sign}(K_s, h_G) \tag{10}$$

Now, we have one part (the signature unit) of the authentication data (denoted as λ). The critical thing is how many hash values are taken as evidence of integrity unit. In another view, which part of the MHT is recorded as the evidence for verification. For example, in Fig. 2, the circled area of the tree contains all VO levels, VOL levels and VOP levels. Thus, all the hash values within the area would be recorded as the integrity unit. Recall that in ULP scheme [7], it is the layer's priority level that proportionally determines its amount of FECs attached to itself.

Our Unequal Loss Verification ULV scheme simulates the policy of ULP by matching the amount of hash values with different priority levels. We define a scheme $\mathcal{ULV} = (\mathcal{M}, \mathcal{T})$ that consists of two functions. The matching function $\mathcal{M}$, given inputs a tree T and objects' encoding priority levels, assigns each node on the tree with a priority level and gets a prioritized tree T_p. The truncating function $\mathcal{T}$, given inputs of a tree T_p and a threshold value θ, truncates all the nodes on the tree whose priority level $p < \theta$, and outputs T'_p. We use T'_p as the final tree and record every hash values of its nodes. For a given virtual sequence VS, we compose its integrity unit as $h_{VS} = \{h_G, L5_j, L4_j, L3_j, \cdots\}$, $j = \{1, 2, \cdots\}$. Thus, we combine them and get the authentication data $\lambda = \{\sigma, h_{VS}\}$.

4.2 Amortizing authentication data

After generating the authentication data, we employ ECC encoders to encode them and amortize them onto the packets before sending them out over an erasure channel (We use the method introduced in [24]). One complex way is to encode different portions of the authentication data with unequal encoding rate (e.g. high priority level yields high encoding rate) and then append the codewords onto the packets. However, the complexity does not take much advantage. For simplicity, we treat the authentication data uniformly with the same encoding rate as of the highest priority layer. The encoding procedure is described as follows: [2]

1. With the systematic ECC algorithm $Enc_{2n-k,n}(\cdot)$, a codeword $X = (h_1, h_2, \cdots, h_n, x_1, \cdots, x_{n-k})^T = Enc_{2n-k,n}(h_1, h_2, \cdots, h_n)$ is produced, where all symbols are in field $GF(2^{w_1})$, n is the number of packets in a group, and k is the minimum number of expected received packets.
2. Dividing the concatenation $x_1 \parallel x_2 \parallel \cdots \parallel x_{n-k}$ into k symbols $y_i \in GF(2^{w_2})$, $i = 1, 2, \cdots, k$. With the erasure code algorithm, a codeword $C_r = Enc_{n,k}(y_1, y_2, \cdots, y_k)$ is produced. Denote the n symbols in the codeword C_r as integrity units $r_1, r_2, \cdots, r_n$.
3. Similarly, dividing signature σ into k symbols σ_i of the same size, $i = 1, 2, \cdots, k$. $\sigma_i \in GF(2^{w_3})$(Note: σ_i and y_i may be of different sizes.) Then encode the signature to produce a signature codeword $C_s = Enc_{n,k}(\sigma_1, \sigma_2, \cdots, \sigma_k)$. Denote the n symbols in the codeword C_s as signature units $s_1, s_2, \cdots, s_n$.

[2] The signature part and integrity part of the authentication data are processed differently to assist explanation of the following operations.

Next, we append integrity unit r_j and signature unit s_j on packet Pj, for all $j = 1, 2, \cdots, n$. That is, the packet Pi now consists of r_i, s_i and $Pi_1, Pi_2, \cdots, Pi_l$.

4.3 Transcoding and transforming

On receiving a stream, a proxy is allowed to do transcoding and transforming operations before retransmission. First, we focus on transcoding. Based on MPEG-4 stream structure, transcoding means that we preserve certain (important) branches of a MHT and truncate other (unimportant) parts. I.e. the shadow part in Fig. 2 could be truncated if necessary. In this example, we discard the subtree $(VO_2 - VO_3)$, keep the subtree root and keep the subtree (VO_1). Apparently, the original authentication data λ has to be changed to a new one λ'. The new data should contain the original signature σ, the new integrity unit h_{VO_1} and the new signature σ_P (signed by the proxy on the root of the subtree $VO_2 - VO_3$, for committing the changes made). We get the new authentication data $\lambda' = \{\sigma, \sigma_P, h_{VO_1}\}$. Using above amortization method, we can append them onto the packets and send them out. [3]

Transforming is another way of adapting to narrow bandwidth, e.g. in Quality of Service network. It simply re-organizes the stream into more but smaller packets. Sometimes enlarging packet size may improve on packet loss rate, which is not supported by transcoding. After transforming, the authentication data must be encoded again to be amortized into the new (larger or smaller) packets. Note that by transforming the packets, the whole stream size will be slightly different from the original size due to more or less packet headers.

4.4 Verifying

The verification process includes unpacking, decoding and verifying, which reverses the generation process. Based on the erasure coding, at least k out of n packets of a group should be received in order to recover the authentication data. Suppose k packets $P_1, P_2, \cdots, P_k$ are received successfully. The integrity units $\hat{r}_1, \hat{r}_2, \cdots, \hat{r}_k$ and the signature units $\hat{s}_1, \hat{s}_2, \cdots, \hat{s}_k$ are recovered from the received packets. With the decoder $Dec_{n,k}(.)$ and $Dec_{2n-k,n}(.)$, the authentication data is reconstructed as $\hat{\lambda} = \{\hat{\sigma}, \hat{h_{VS}}\}$. Then, the signature can be verified with algorithm $\mathsf{Veri}(K_p, \hat{\sigma}, \hat{h}_G)$, where K_p is the public key of the producer. If $\mathsf{Veri}(.)$ is true, then continue to verify the integrity unit; if not, the object group is bogus and discarded. To this end, the client reconstructs the hash tree h'_{VS} according to formulas (4)-(8). The extracted integrity unit $\hat{h_{VS}}$ is now compared with the constructed unit h'_{VS}, which is actually the comparison of two MHTs. If there is no transcoding operation, we require $h'_{VS} = \hat{h_{VS}}$ for successful verification. If there are transcoding operations, the signature and integrity units are verified one by one in descendent order.

[3] The new stream size shrinks since both the size of stream data and authentication data are reduced.

5 Security and performance analysis

The security of our scheme relies on the security of the Merkle hash tree. Fortunately, Merkle hash tree has very nice security properties [8]. In this analysis, we focus on how much of a meaningful stream is verifiable. Then, we analyze the computational cost of each role in the scheme.

5.1 Verification probability

Recall the ULP scheme in [7], the fundamental layer has been assigned the most FECs to resist the heaviest packet losses. A stream is not successfully received if even its base layer is not correctly recovered. In this condition, no verification is available. In our scheme, we attach the same number of parity units for authentication data as of the base layer. Assuming the base layer is recoverable, the authentication data is also recoverable. Additionally, assuming an erasure channel with independent packet losses, given ρ the packet loss probability. The group of n packets transferred over the erasure channel may have probably $\binom{n}{k}\rho^{n-k}(1-\rho)^k$ packets received. The verification delay for a group of n packets is $O(n)$. In our definition, only those recovered content of a received stream can be verified. If we receive enough packets to recover only the base layer, we are able to verify it from the authentication data. We say that only the base layer is verifiable. Surely, more packets received mean that more contents are verifiable. In other words, the rate of the reconstructed hash tree T' over the recovered hash tree $\hat{T}$ from the received packets directly determines the verification rate $T'/\hat{T}$ over the object group, given the signature on $\hat{T}$ is valid.

5.2 Computational cost

We study the computational cost related to security and ignore object encoding/decoding. Firstly, in case of no transcoding operation, the signature is generated once for a group of packets. The computational cost for signature generation and verification depends on the signature scheme selected. However, the signature verification can be much faster than signature generation. I.e. for a RSA signature scheme, the verification time can be only 4% of the signature generation time (with the public exponent equals 17). Secondly, based on equations (4)-(9), verifying the integrity unit depends on how many hash operations are required in generating the hash tree. For a MHT with n data items, the total number of hash operations is roughly $2n$.

Last, when there are transcoding operations, the cost of generating and verifying the signature is proportional to the number of transcoding operations. By this we assume one transcoding operation produces one signature generation/verification operation. However, for signature generation, the cost at the producer is fixed by one; the cost at proxy is proportional to the number of signature operations. For signature verification, the final receiver has to verify all the signatures, but at relatively lower cost. The receiver will also spend time on reconstruct the (probably partial) hash tree over the object group. A partial

hash tree means that the receiver spend less time on constructing the tree at the cost of verifying more signatures.

6 Publishing the stream in p2p CDN

We introduced the ULV scheme above, note that in section 4.2, the authentication data units are amortized into the packets. While the packets are transmitted over a lossy network, this method makes sure that the authentication data can be recovered from the received packets. In S-R model, the scheme runs well. But in S-P-R model, there is a subtle problem. Suppose the proxy receives t out of n packets ($t < n$, t is enough to recover some base layers, but not enough to recover other layers), the proxy faces a dilemma: whether to transfer those t packets unchanged to the receiver (which is simple and saves computation time, but wastes bandwidth) or to transcode the stream (which wastes computation resources and needs less bandwidth). Another way-transforming, although feasible, is also not practical. In either case, the amortizing method, similar to packet based schemes, suffers either from high computation or from wasting bandwidth. In p2p CDN, there exists another reliable way to transfer the data stream and its authentication data separately. Note that the size of authentication data is much smaller than that of data stream proportionally.

6.1 Publishing and retrieving

Suppose a stream S is divided into n virtual stream object groups (VSs): $S = \{VS_1, VS_2, ..., VS_n\}$. According to equations (9,10), we compute authentication data $\Lambda = \{H_S, \Sigma\}$, where $H_S = \{h_{VS_1}, h_{VS_2}, ..., h_{VS_n}\}$, and $\Sigma = \{\sigma_{VS_1}, \sigma_{VS_2}, ..., \sigma_{VS_n}\}$.

Given the name of the stream S as $N(S)$, the publisher inserts the stream content S to the p2p overlay with key $h[N(S), -]$ and inserts the stream authentication data Λ with key $h[N(S), A]$.

To retrieve the stream, the user first computes the two keys $h[N(S), -]$ and $h[N(S), A]$. Then she looks up the keys in the p2p CDN and expects to get some storing locations of the stream. The user then sends requests to the selected storing points and downloads the stream as well as its authentication data.

Suppose the stream is extremely large, it is easy to publish it individually under different look-up keys. For example, VS_i can be published using the key $h[N(S), i]$, ($i \in \{1, 2, ..., n\}$). In this case, the key $h[N(S), -]$ may be mapped to a description file of the stream (i.e. a README file on how to download the stream). By retrieving the description, the user proceeds in downloading by iteratively querying with $h[N(S), i]$, until she downloads the whole stream. Since every virtual object group VS_i is downloaded separately, if the authentication data was downloaded first, it can also be verified individually. Using this first-come-first-serve method, we achieve the on-the-fly verification based on VSs.

6.2 Republishing and retrieving

We assume initially that the stream can be transcoded or transformed by any intermediate proxy. Since the scheme is actually transparent over packet level, transforming operation doesn't change anything and needs no republishing. However, transcoding operation changes the content by removing some layers, thus it needs to be published again. For instance, suppose the original stream S is played in real time under wide bandwidth 10MB/s. S can be transcoded to $S' = \{VS'_1, VS'_2, ..., VS'_n\}$ to meet certain narrow bandwidth 1MB/s (with low quality). The proxy needs to compute new lookup keys for S'. Without loss of generality, assume the new stream name is $N(S')$ [4]. The proxy now insert S' into the CDN with key $h[N(S'), -]$. Note that the authentication data Λ needs no republishing if it covers the layers being removed. The retrieving process is similar with above method. To verify the transcoded stream, all content of the stream are able to be verified except for the removed layers.

6.3 An interactive verification scheme

The authentication data Λ is required to be published/retrieved together with stream data S. If the user would not waste their bandwidth for retrieving it, but still wants to check the data randomly, she can choose an interactive way of verifying. In this case, there must be an online verification server who holds Λ and answers arbitrary queries in real time. Considering Λ as MHT, the server, being queried with a leave, may answer with a path (with length log(n)) from the leave to the root of MHT. This is a bandwidth-storage wise solution. Our basic scheme is flexible on verifying various portions of the group structure. The scheme can be flexible in more ways.

7 Prior works and discussions

In the literature, there existed a bunch of research works [15–24] that focused on multicast authentication and stream authentication. We analyzed P-SASs in section 2. In case of erasure channel or multicast channel, the schemes works well by passing packets through routers. However, as we have indicated, the packets have to be manipulated while being passed through proxies. The previous schemes don't allow packet modification once the producer's finishing the preparation phase. In one word, none of them can work under S-P-R model. One recent work [26] uses distillation code to defend against pollution attacks over a polluted erasure channel. Another recent work, [27], deals with the same problem under a fully adversarial model. In our threat model, we only assume the erasure channel. Since the mechanisms of [26, 27] works on the encoding/decoding

[4] One way of linking the new name with the original one is to put the new name in a new README file and published with key $h[N(S), new]$. In fact, any searching engine can help derive the new name, which is beyond the scope of this paper.

phase, we can adapt the coding mechanism into our scheme to defend pollution attacks (this is our future work).

The current approach, [25], describes an on-the-fly verification scheme on transferring huge file over p2p CDN. It addresses mainly the on-the-fly property on verifying small blocks while being delivered over an erasure channel. Although the intermediate operations are not considered in their work, it has a similar publishing method as ours.

One possible thing needed to be pointed out of our scheme is the potential abuse of content copyright, since we allow intermediate content modification. While traditional Digital right Management (DRM) systems focus on end-to-end protection, the future DRM systems must consider the protection of content in-delivery. Thus, we assert that our scheme does not apparently violate the DRM framework, but enriches it. How to ensure the content's intellectual property, at the meanwhile to provide high flexibility is still a challenging topic.

8 Conclusions and future directions

We proposed an ULV scheme that can verify a stream flexibly. The scheme is also easily extensible and scales well, but relies on special stream format (e.g. MPEG-4). We elaborated how the scheme works in S-R model and how it works in S-P-R model. Both are under the assumption of "erasure channel", but can be adapted to "polluted erasure channel", e.g. by using distillation code [26]. The scheme is also enriched by imaginative appliances like multi-source stream authentication. Our analysis shows that it is secure and cost-effective. In the near future, we will develop it within MPEG-4 framework and apply it into interesting applications. More experiments need to be done for testing its performance.

References

1. ISO/IEC 14496-1:2001 Information Technology - Coding of Audio-Visual Objects-Part 1: Systems
2. ISO/IEC 14496-2:2003 Information Technology - Coding of Audio-Visual Objects-Part 2: Visual
3. Weiping Li, *Overview of fine granularity scalability in MPEG-4 video standard,* IEEE Trans. on Circuits and Systems for Video Technology, 11(3):301-317, Mar 2001
4. Gerald Kuhne, Christoph Kuhmnch, *Transmitting MPEG-4 Video Streams over the Internet: Problems and Solutions,* ACM Multimedia, 1999
5. Chong Hooi Chia, M. S. Beg, *MPEG-4 video transmission over bluetooth links,* IEEE International Conference on Personal Wireless Communications, 280-284, 2002
6. P. Ikkurthy, M.A. Labrador, *Characterization of MPEG-4 traffic over IEEE 802.11b wireless LANs,* 27th Annual IEEE Conference on Local Computer Networks, 421-427, 2002
7. A. E. Mohr, E. A. Riskin, R.E Ladner, *Unequal loss protection: graceful degradation of image quality over packet erasure channels through forward error correction.* IEEE Journal on Selected Areas in Communications, 18(6):819-828, 2000

8. R. C. Merkle, *A certified digital signature*, Crypto'89, Lecture Notes on Computer Science, Vol. 0435, pp. 218-238, Spriner-Verlag, 1989.
9. Jonathan Robbins *RTT and Loss vs. Packet Size and Bitrate.* http://www.cs.unc.edu/ robbins/comp249/HW3/
10. S. Saroui, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy, *An analysis of Internet content delivery systems,* in Proc. 5th Symposium on Operating Systems Design and Implementation (OSDI), Boston, MA, Oct. 2002.
11. M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann, *Practical loss-resilient codes,* in Proc. 29th Annual ACM Symposium on Theory of Computing (STOC), El Paso, TX, May 1997.
12. M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, A. Singh, *Splitstream: High-bandwidth multicast in a cooperative environment,* Proc. 18th ACM symposiom on operating systems principles (SOSP), NY, Oct. 2003.
13. V. N. Padmanabhan, H. J. Wang, and P. A. Chou *Resilient Peer-to-Peer Streaming* IEEE ICNP'03, Atlanta, GA, USA November 2003
14. R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, *Multicast security: A taxonomy and some efficient constructions,* in Proc. IEEE INFOCOM'99, New York, NY, 1999.
15. R. Gennaro and P. Rohatgi, *How to sign digital streams,* in Advances in Cryptology-CRYPTO'97, Santa Barbara, CA, Aug. 1997.
16. C. K. Wong and S. S. Lam, *Digital signatures for flows and multicasts,* in Proc. IEEE International Conference on Network Protocols, Austin, TX, Oct. 1998.
17. P. Rohatgi, *A compact and fast hybrid signature scheme for multicast packet authentication,* in Proc. 6th ACM Conference on Computer and Communication Security (CCS), Singapore, Nov. 1999.
18. A. Perrig, R. Canetti, J. D. Tygar, and D. Song. *Efficient authentication and signature of multicast streams over lossy channels.* In Proceedings of the IEEE Symposium on Research in Security and Privacy, pages 56-73, May 2000.
19. A. Perrig, R. Canetti, D. Song, and J. D. Tygar. *Efficient and secure source authentication for multicast.* In Proceedings of the Symposium on Network and Distributed Systems Security (NDSS 2001), pages 35-46. Internet Society, Feb. 2001.
20. P. Golle and N. Modadugu, *Authenticated streamed data in the presernce of random packet loss,* in Proc. NDSS01, San Diego, CA.
21. S. Miner and J. Staddon. *Graph-based authentication of digital streams.* IEEE S&P 2001, pages 232-246.
22. A. Perrig. *The BiBa one-time signature and broadcast authentication protocol.* In Proceedings of the Eighth ACM Conference on Computer and Communications Security (CCS-8), pages 28-37, Philadelphia PA, USA, Nov. 2001.
23. J. M. Park, E. K. Chong, and H. J. Siegel. *Efficient multicast packet authentication using signature amortization.* IEEE S&P 2002, pages 227-240.
24. A. Pannetrat and R. Molva, *Efficient multicast packet authentication,* in Proc. NDSS'03, San Diego, CA.
25. Maxwell N. Krohn, Michael J. Freedman, David Mazires *On-the-Fly Verification of Rateless Erasure Codes for Efficient Content Distribution.* IEEE S&P'04, California, USA.
26. C. Karlof, N. Sastry, Y. Li, A. Perrig, and J. Tygar, *Distillation codes and applications to DoS resistant multicast authentication,* in Proc. NDSS'04, San Diego, CA, Feb. 2004.
27. A. Lysyanskaya, R. Tamassia, N. Triandopoulos, *Multicast Authentication in Fully Adversarial Networks.* IEEE S&P'04, California, USA.